P515716.PDF [Page: 1 of 57]

Image Cover Sheet CA011071

CLASSIFICATION	SYSTEM NUMBER
UNCLASSIFIED	515716

TITLE

Plans and goals in a testbed for inteligent aiding

System Number:

Patron Number:

Requester:

Notes:

DSIS Use only:

Deliver to:

This page is left blank

This page is left blank

CA011071

DOCUMENT REVIEW PANEL PUBLICATION RECORD

A. BIBLIOGRAPHIC DATA	
1. PERFORMING AGENCY(IES) Artificial Intelligence Management and Development Corporation, 206 Keewatin Avenue, Toronto, ON M4P 1Z8 CANADA	2. CONTRACT and/or PROJECT NO PW&GS W7711-8-7564/001/TOR; 3ad13
3. SPONSOR (DND Project Officer or Directorate)	4 PUBLICATION SERIES and NO.. Contract Report CR 2001-011; AC203
5. TITLE : (U) Plans and Goals in a Testbed for Intelligent Aiding - Final Report	
6. PERSONAL AUTHORS . Edwards, Jack L.	7 DATE OF PUBLICATION · July 1 , 2000
8. PUBLISHING AGENCY (Name of Establishment) DCIEM	
B. SECURITY CLASSIFICATION / LIMITATION INFORMATION	
9a Overall **SECURITY Classification** or **DESIGNATION** of document: UNCLASSIFIED	9b DOCUMENT REVIEW DATE January 1 ,
10a. **OFFICIAL WARNING TERM** (e g. Canada/US Eyes Only)	
10b **Reasons for Classification or warning term:**	

11a. DETAILS OF FOREIGN CLASSIFIED INFORMATION

Country of Origin	Highest Level	PAGES ON WHICH CLASSIFIED OR DESIGNATED INFORMATION IS CONTAINED		
		Text	Tables/Figures	Classified Titles Cited

11b. FOREIGN CLASSIFIED REFERENCES	
12. DOCUMENT RELEASABILITY Unlimited distribution	13. APPROVED COUNTRIES
14 DOCUMENT ANNOUNCEMENT Unlimited	
15. REASON FOR NO ANNOUNCEMENT	
D. AUTHORIZATION	
16. Meeting number and date of action of Establishment or HQ Document Review Panel. 1-2001, 2001/02/21 DRP Meeting No. Date Approved by :	

DCIEM No. CR 2001-011

PLANS AND GOALS IN A TESTBED FOR INTELLIGENT AIDING – FINAL REPORT

by

Jack L. Edwards

Artificial Intelligence Management and Development Corporation
206 Keewatin Ave., Toronto, ON
M4P 1Z8

PWGSC Contract No. W7711-8-7564/001/TOR

on behalf of
DEPARTMENT OF NATIONAL DEFENCE

As represented by
Defence and Civil Institute of Environmental Medicine
1133 Sheppard Avenue West
Toronto, Ontario, Canada
M3M 3B9

DCIEM Scientific Authority:
Name: K. Hendy
Telephone no. (416) 635-2074

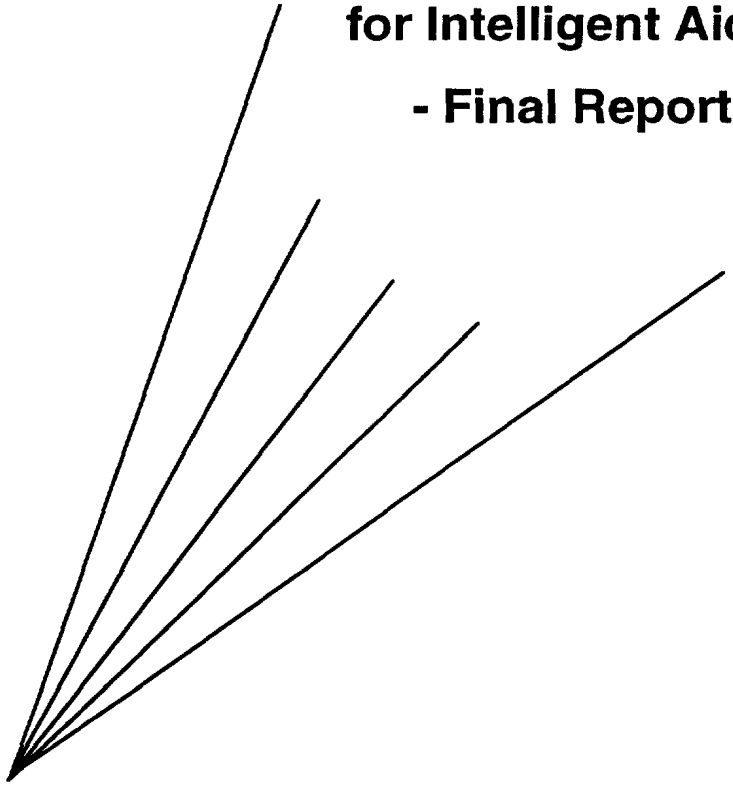
1 July 2000

Prepared for: Public Works and Government Services
Prepared by: Artificial Intelligence Management and Development Corporation
AIM (AC203, July, 2000)
Contract: W7711-9-7564/001/TOR
"Plans and Goals in a Testbed for Intelligent Aiding"
Scientific Authority:
Mr. Keith C. Hendy
Simulation and Modelling for Acquisition, Rehearsal and Training (SMART)
Defence and Civil Institute of Environmental Medicine

© 2000 Her Majesty the Queen in right of Canada, as represented by the Minister of National Defence

Plans and Goals in a Testbed for Intelligent Aiding

- Final Report -



Artificial Intelligence
Management and Development Corporation
206 Keewatin Ave., Toronto, Ontario M4P 1Z8

Table of Contents

Abstract iv
Background 1
Research Approach 2
Study Objectives 3
Extending Action, Goal and Plan Recognition in the
LOCATE Workspace Layout Tool 4
Requirements for a General Plan Recognition System Within *LOCATE* 8
Plan Recognition and Information Processing/Perceptual Control Theory (IP/PCT) 10
The Notions of Plans 12
The Nature of Hierarchies 13
Hierarchies and Plans 15
Hierarchies Revisited 17
Shared Concepts for Intelligent Adaptive Interfaces 18
Generalising Plan Recognition Principles to Other Projects at DCIEM
Integrated Performance Modelling Environment (IPME) 19
Safework® 20
Incorporating Planning Into a Standalone Pluggable Intelligent Aiding Module 21
References 24

APPENDICES
Appendix A: Outline Version of *LOCATE's* Goal and Plan Hierarchy A-1
Appendix B: Annotated List of *LOCATE* Source Code Files A-11

FIGURES

Figure 1. A Portion of *LOCATE's* Plan Hierarchy 5
Figure 2. Conventional Goal and Associated Goal-Plan Hierarchies for Selecting a Workstation 6
Figure 3. Conventional Goal and Associated Goal-Plan Hierarchy for Opening a Workstation's Attribute Window 7

TABLES

Table 1. Goal Interactions 18

P515716.PDF [Page: 9 of 57]

Abstract

The work of this contract extends efforts on an intelligence testbed using the *LOCATE* Workspace Layout Tool. Work on plan recognition aspects of the testbed was extended in two ways: first, by creating an action, goal and plan hierarchy that permits *LOCATE* to match user actions to possible hierarchies, and second, by developing a generalisation of the goal and plan hierarchy through a decomposition into two-level, hierarchical recipes, which can be mixed and matched from various actions at the interface. Other work included comparing concepts used in the AI planning work in *LOCATE* to similar concepts in an Information Processing/Perceptual Control Theory (IP/PCT) model being developed at DCIEM; similarities and differences of the two approaches were identified and discussed. Ideas for generalising *LOCATE's* planning principles to development projects such as IPME and Safework®, projects supported by DCIEM, were explored. Finally, the concept of building an interface analyser, to help automate customisation of intelligent aiding for other applications, was described and discussed.

Sommaire

Le travail régi par le présent contrat continue les efforts fournis pour un banc d'essai d'intelligence utilisant l'outil d'aménagement de l'espace de travail, LOCATE. Les travaux couvrant les aspects de reconnaissance de plan touchant le banc d'essai ont été accrus de deux manières : premièrement, en établissant une hiérarchie d'action, de but et de plan, qui permet à LOCATE de jumeler les actions des utilisateurs à des hiérarchies possibles et, deuxièmement, en procédant à une généralisation de la hiérarchie de plan et de but par une division hiérarchique à deux niveaux des recettes, qui permettent un choix de combinaisons générées par différentes actions à l'interface.

Notre travail consistait entre autres à comparer les concepts utilisés dans le travail de planification IA de LOCATE avec des concepts similaires dans un modèle de traitement de l'information/théorie perceptuelle de commande (TI/TPC) qui est en train d'être mis au point à l'IMED. Les similarités et les différences des deux approches ont été déterminées et ont fait l'objet de discussion. On a étudié à fond les idées qui visent à appliquer les principes de planification du LOCATE à l'ensemble de projets tels que l'IPME® et le Safework®, projets qui sont appuyés par l'IMED. Enfin, l'idée de construire un analyseur servant à personnaliser l'aide intelligente pour les autres applications a été décrite et a fait l'objet de discussion.

Executive Summary

The purpose of the work reported here was to develop generalizations from principles and techniques of intelligent aiding that will be useful to projects developed at Canada's Defence and Civil Institute of Environmental Medicine (DCIEM).

In realizing that purpose, the principles and techniques of intelligent aiding implemented in DCIEM's LOCATE Workspace Layout Tool were combined with other, similar ones from related areas of research. LOCATE is a CAD tool that allows users to create designs and analyze their communication efficiency.

The LOCATE tool is serving as the focus for discussion, design and implementation in an emerging testbed for elaborating ideas on intelligent aiding. Earlier work on building an infrastructure to support such aiding in LOCATE was extended so that the tracking of all low-level user actions is now complete, as is the ability of LOCAT to infer the goals those actions imply.

Further extensions during this work phase included the identification and elaboration of categories of actions that imply a need for help. A category of "Non-Responsive Actions," or actions for which there is no system response, was selected for implementation. Help was designed and implemented first of all to inform users that such actions produce no response and second, to help them identify and achieve whatever goal(s) they had in mind when they performed those actions.

Task and help, actions and goals are represented in explicit models of task, user and system in LOCATE, and are displayed in associated windows in its interface. In the Task Model, an action and goal history is maintained as the user creates and analyzes designs. As that word proceeds, help of various kinds is initiated by LOCATE and, in a similar way, its System Model maintains an actions and goal help history.

Several projects were identified as likely candidates for the application of the principles and techniques identified here, but none was at a stage where that could be done in the short term. Those and other projects will continue to be monitored for opportunities to incorporate intelligent aiding capabilities.

Background

Recent work at DCIEM (W7711-8-7476; W7711-8-7480) has focused on adding intelligent aiding capabilities to the *LOCATE* Workspace Layout Tool, a specialised CAD system for creating and analysing workspace (facility) layout designs (Hendy, 1984, 1989).

Principles inherent in that work are applicable to a much wider range of software, which potentially includes other development projects at DCIEM. Preliminary work to explore and expand those principles and to develop generalisations that might apply to such efforts has been completed (W7711-9-7546).

DCIEM extended those efforts in the current contract by:

- examining how goal recognition, developed for the *LOCATE* workspace tool, might be incorporated into more general and more complex plan recognition, a well-developed area of research in Artificial Intelligence (AI);
- exploring how plan recognition might be accommodated within an Information Processing/ Perceptual Control Theory (IP/PCT) model (Hendy, 1998; Hendy, East, & Farrell, 1998; Hendy, Liao, & Milgram, 1997);
- implementing aspects of plan recognition into the *LOCATE* tool;
- exploring how principles of plan recognition might be generalised to other projects;
- examining how plan recognition might be incorporated into a standalone pluggable intelligent aiding module useful to a variety of software development efforts.

The primary focus of the work here was an extension of DCIEM's intelligent aiding testbed through a deeper understanding of plan recognition and its relationship to goal inference. The work employed the *LOCATE* Workspace Layout Tool as a context for exploring and implementing intelligent aiding concepts and techniques. As that work progresses, the intent is to add other practical tools to the testbed that are related to *LOCATE*.

Research Approach

The research approach taken in this project included reviewing techniques for plan recognition in AI and related areas (Edwards & Sinclair, 2000; Küpper & Kobsa, 1999; Leash, Rich, & Sidner, 1999) and comparing that work to the work on goal inference used within the *LOCATE* Workspace Layout Tool.

The research focused on how the testbed work with *LOCATE* could be extended to handle the more complex and general notion of plan recognition. The ultimate goal was to understand more fully what the user is doing at any given moment and consequently to provide intelligent aiding in support of the task the user is performing.

Another goal of the research was to integrate the principles and procedures into a coherent theory of information processing and control. The approach examined how concepts related to goal and plan recognition might be described and possibly accommodated within an Information Processing/Perceptual Control Theory (IP/PCT) model (see Powers, 1999; 1973; Robertson & Powers, 1990)

As the work on theory progressed, suggestions for how plan recognition could be incorporated within the *LOCATE* tool were explored and some implementation work to illustrate how that might be done was conducted.

In a previous contract (W7711-9-7546), work was done on how principles of goal inference might be generalised to provide intelligent aiding to development efforts within DCIEM, beyond the *LOCATE* project. Three areas of potential applicability were identified: 1) the F-18 Incremental Modernisation Program; 2) the Tactical Battlefield Command System; and, 3) the Navy Command and Control System. The work on generalisation in this contract was extended to plan recognition and focused on application developments more closely related to *LOCATE* itself. Work on standalone shells or pluggable modules as well as integrated intelligent aiding systems were of particular interest.

Similar to work in the aforementioned contract, a key concern of this effort was determining how to create an intelligent aiding component that is at least partially decomposable from the software it supports. The added value for that effort will be an understanding of how plan recognition can be implemented within such a module.

LOCATE is a mature tool that is being used in practical applications. Its role in DCIEM's continuing efforts to build a testbed for intelligent aiding is to provide a solid context within which ideas and principles can be tested. In the current project, it served to facilitate an understanding of how goal and plan inference can support intelligent aiding for software users.

STUDY OBJECTIVES

Study objectives for this project were:

- to examine how the principles used in goal recognition with the *LOCATE* workspace layout tool can be extended to the more complex process of plan recognition;
- to identify requirements for adding aspects of plan recognition to *LOCATE*;
- to implement some select examples of plan recognition based on those requirements;
- to explore how goal–plan recognition, as developed within AI and related areas, might be accommodated within an Information Processing/Perceptual Control Theory (IP/PCT) model;
- to explore how plan recognition might be generalised to other software development projects of interest to DCIEM;
- to examine how to incorporate aspects of plan recognition into a standalone pluggable intelligent aiding module useful across a variety of development efforts.

Extending Action, Goal and Plan Recognition in the *LOCATE* Workspace Layout Tool

LOCATE's action, goal and plan recognition capability was examined to determine how it might be extended to the more complex and general process of plan recognition.

Currently, *LOCATE* tracks actions and infers goals and plans for both user and system tasks. The focus in this report was on the actions, goals and plans associated with those tasks. To build a more general process of goal and plan recognition that might be useful to other applications, a complete hierarchy of task related activities was generated for *LOCATE*.

The hierarchy is composed of a variety of nodes that contain variables, which are instantiated with values that depend on the context of a user's actions. For example, when a user wishes to create an object in the design space, he or she typically clicks (or double-clicks) on the icon for that object in the *LOCATE* tool palette. The specific palette object provides the value for the instantiated action and inferred goal.

More specifically, the intentional node in the plan hierarchy that relates to that action is "To select a tool <T> in *LOCATE's* tool palette." As there are some two dozen tools in the palette, the value of the object in the node can be any one of those different objects.

Once an object has been created, actions are defined with respect to the particular instantiation and other nodes in the hierarchy represent the actions, goals and plans inferred from that action.

The hierarchy was developed using a graphical tool for representing a variety of types of graph and chart. As the graph of the hierarchy is large, even a small portion is impossible to display in a document such as this. It was possible, however, to convert the graph to an outline form, which is included as Appendix A. An excerpt of the graph itself appears in Figure 1,[1] which shows one way of changing attributes of objects in a *LOCATE* workspace, specifically workstation objects. There is an optional selection of an attributes tab on the workstation window, which is selected when the window is open and another tabbed section is being displayed. The tabs for the other two sections of this window relate to data entry for Link Functions and Priority Weights.

---

[1] The entire graph is considered part of this report and has been delivered in electronic form.

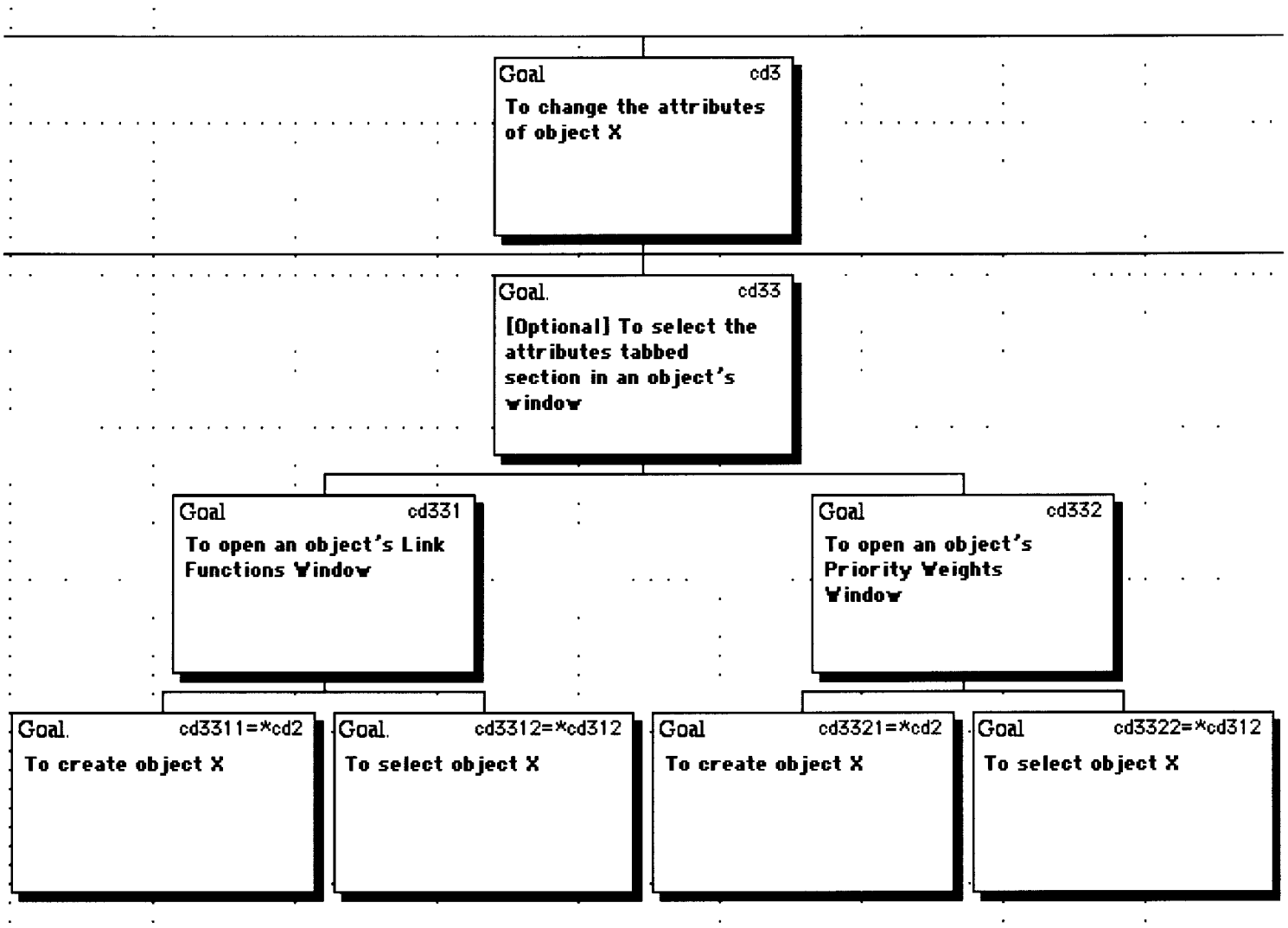


Figure 1. An Excerpt of *LOCATE's* Action, Goal and Plan Hierarchy

P515716.PDF [Page: 16 of 57]

The codes in the upper right corner of each of the hierarchical goal units are unique identifiers. The codes followed by an equals sign (=) are pointers to other units (preceded by an asterisk (*)) and refer to elements in the hierarchy that the current unit duplicates. This type of referencing is a qualification on the notion of hierarchy as it is used here. The hierarchy is more accurately described as a directed graph rather than a strict tree structure, since it allows for a a node with more than one parent.

LOCATE provides windows on its explicit models of task, user and system. Those windows are used to display content that shows each action performed by a user and the goals that *LOCATE* infers from those actions. Although the representations displayed in those windows include partial plans, also inferred from user actions, a more complete way of representing a user's plans was needed. Consequently, a new window was added that alows for the tracking and representations of user plans within *LOCATE's* complete goal and plan hierarchy.

In Figure 2, the intended goals matched by the current user action appear in the left portion of the window. These may be characterised as "Conventional Goals" since they reflect a level of the goal and plan hierarchy that is considered to be a conventional way of describing the intent of the user action.

Clicking on any one of the matched goals displays an associated plan. In the example in the figure, the goal (intention) selected is "to rotate an object." The associated plan in which it participates appears on the right with the goal itself (represented as a subgoal in a broader expression of an inferred plan) appearing as the fourth subgoal from the bottom.

As can be seen from the items in the window, clicking on a workstation invokes a number of possible goals and *LOCATE* cannot be sure, without further action on the part of the user, which one of those goals the user is pursuing. As a hypothetical case in point, assume that the user's next action is the selection of the menu item, "Workstation" from *LOCATE's* View Menu, which brings up the attributes portion of the workstation's window. That action reduces the number of inferred goals to one (see Figure 3). Again, the goal appears both in the left and the right panels of the window. In the former, it is now the only goal and, in the latter, it is the second from the bottom.

P515716.PDF [Page: 18 of 57]

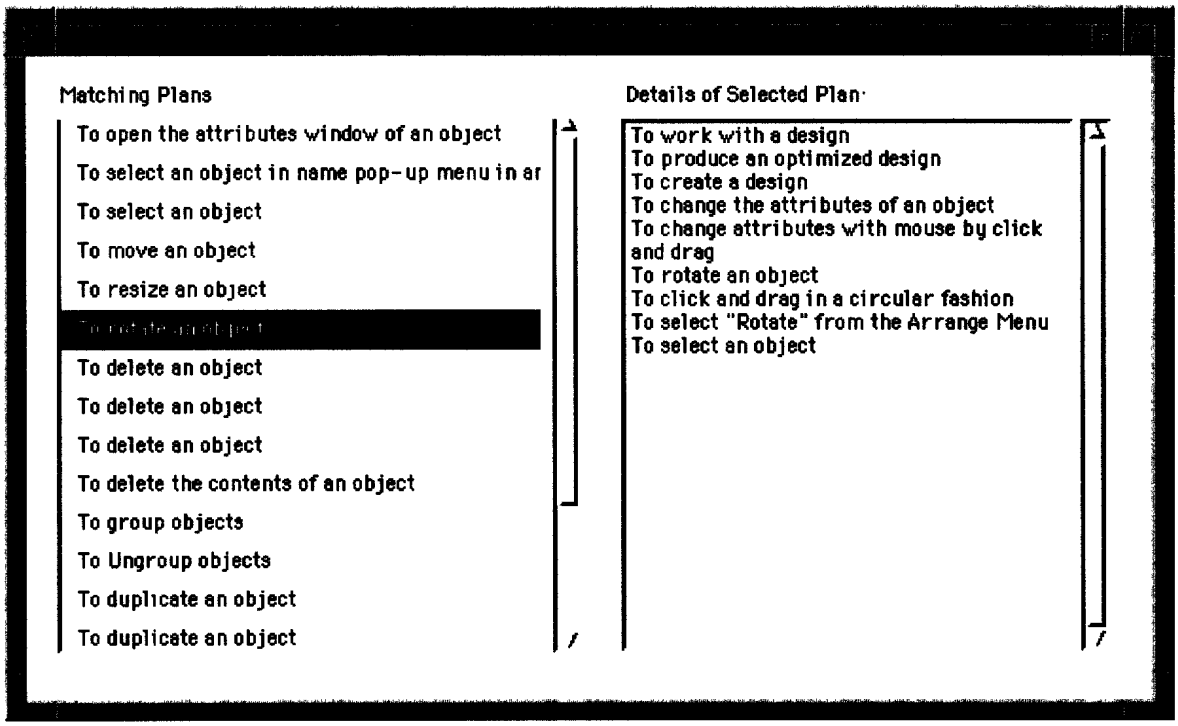


Figure 2. Conventional Goal and Goal-Plan Hierarchies Matched When Selecting a Workstation

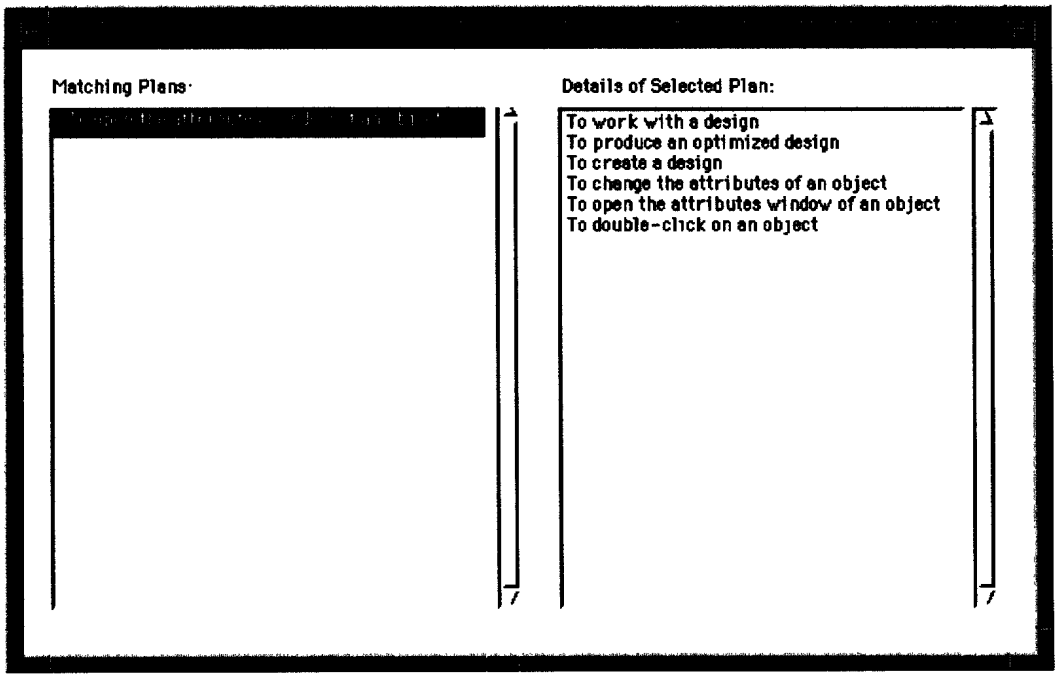


Figure 3. Conventional Goal and Goal-Plan Hierarchy
Matched When Opening a Workstation's Attribute Window

Requirements for a General Plan Recognition System Within *LOCATE*

Tracking user actions and associating them with the goals a user might be pursuing was a first step in building a plan recognition system for *LOCATE*. That work was based on a preliminary analysis of all of the possible actions a user might perform while using the *LOCATE* Workspace Tool.

The concept was extended in the current contract by organising those actions and goals into a hierarchical representation, which now resides in *LOCATE* itself and provides the application with a view on how high level goals can be achieved by pursuing alternate lower-level subgoals, the latter terminating in user interface actions.

Further, the goals inferred from a user's action and represented in *LOCATE's* task model reflect conventional goals, that is, goals that would normally be inferred from a user's action by an observer familiar with *LOCATE*. Those goals are now embedded in the hierarchy of goals and subgoals and provide a broader, more coherent picture of user activity, as illustrated in the two figures in the previous section.

When a user performs some action in *LOCATE*, an algorithm now matches the action to possible goals and plans in the hierarchy that the actions imply. Again, as seen above, those actions are sometimes ambiguous with respect to the user's goal and associated plan(s); other times, a single action will define clearly the goal the user is trying to achieve as well as the means by which he is trying to achieve it.

This approach to plan recognition is adequate for application domains such as that of *LOCATE*, where many actions and their associated goals and plans are possible, but where the number of alternatives is still manageable. That characteristic means that plan recognition in *LOCATE* is unlikely to encounter a combinatorial explosion of possibilities often characteristic of planning domains.

A further extension to *LOCATE* made its plan recognition capabilities even more general than the approach just described. Following work by Lesh, Rich and Sidner (1999), among others, *LOCATE's* hierarchy was decomposed into a set of component recipes. Those recipes are successive, two-level decompositions of the hierarchy, which can be combined in various ways when attempting to match a given instantiation of a goal and plan hierarchy with a user action.

P515716.PDF [Page: 20 of 57]

Although this approach was implemented in the current version of *LOCATE*, its representation does not differ from the one already described for the hierarchy. Future work will provide distinct displays along with performance comparisons for these and possibly other approaches. This is consistent with the context of an intelligence testbed, where one of the goals is to compare alternative methods of intelligent aiding and combine the most effective aspects of each into an system for further refinement.

An annotated list of the *LOCATE* source code files, which support the current version and incorporate the new hierarchy for plan recognition and two-level hierarchical recipes, appears in Appendix B.

Other work that will help extend the generalisability of *LOCATE's* plan recogniser is a representation of the context of user actions; in other words, the maintenance of a world model in which user actions take place. Part of that work will be to add two new elements for each goal in the system: a set of preconditions, which must hold true before a goal can be pursued, and procedures for adding and subtracting facts from that world model, as a consequence of pursuing and achieving the goal.

Still other work will address issues of goal conflict and cooperation, including attempts to pursue mutually exclusive goals, internal to the system, or coordinate mutually supportive goals among several different human users, or among human users and software systems. The latter becomes important in the context of task allocation, which *LOCATE* will provide examples of in future.

The concepts of preconditions, adding and subtracting facts from a world model and goal interactions are addressed again in the next section.

Plan Recognition and Information Processing/ Perceptual Control Theory (IP/PCT)

The concepts that make up Perceptual Control Theory, taken individually, are not unique. They appear in many theories, some that have been around for a very long time. What makes the contribution unique is the combination of those concepts into an approach that promises to add important insights to the analysis, design and implementation of intelligent, adaptive systems.

It is difficult to say how long the notion of purposeful behaviour has been around but it is discussed early in the life of psychology by William James, among others, later in elaborated theories of learning, and still later in cognitive psychology and information processing theories. More recently, various subfields in artificial intelligence (e.g., the subfield of planning) deal with purposeful behaviour.

Many concepts used in the description, explanation and prediction of purposeful behaviour overlap these disciplines; concepts like **goals**, **plans**, **states**, **feedback**, **hierarchies**, **closed loops**, **feedback loops**, **conflict**, **cooperation**, and so on. The different ways these concepts are defined, elaborated and used can provide useful cross-disciplinary insights.

In the interest of building intelligent, adaptive systems, it is worthwhile to take some time to examine how some of those concepts are treated and what other disciplines have to say to PCT about how they might be used. In particular, PCT is treated here from the perspective of some of the concepts used in AI for building intelligent, adaptive systems. Aspects of its purpose include clarifying differences in the use of same or similar terms, providing a basis for transforming usage between the two fields, identifying apparent discrepancies, highlighting areas where questions remain and laying a foundation for future refinement and elaboration of ideas necessary to building those systems.

PCT's basic unit of purposeful behaviour is the **reference signal**, taken from control theory. Unlike control theory, the reference signal is not given from the outside by some observer or user of the control system but emerges from within. For humans, reference signals are inferred and cannot be observed directly.

Reference signals exist at many different levels of abstraction and are linked together in ways that can help define all complex systems. At any given level of abstraction, a

reference signal is compared to a **perceptual signal** (at that same level) and, if a difference is detected, the system produces responses intended to change the perceptual signal in ways that will reduce that difference. The results of those changes provide feedback in the form of an altered perceptual signal that moves through the same process again. All of this happens in a closed loop system not easily described in terms of traditional notions of cause and effect.

In AI, in the area of planning for example, the notion of **goal** is a key concept. In general, a goal is a **state** and there two states of particular importance in plan generation work: a **current state** of the world and a **desired state** of the world, both held by some entity (person or machine). In a similar way to the negative feedback system in PCT, the object of planning is to reduce the difference between the current state and the desired state and, in the case of AI, this is done through the application of operators. Early work in this area, conducted by Newell and Simon (1963; 1972), made use of a means-ends analysis in the application of operators to reduce that difference.

During this process, it is important to understand the **preconditions**, that is, the conditions for applying an operator, and to make sure that they are met before attempting to apply the operator. A second aspect of the process is understanding the **effects** that will be produced in the world when the operator is applied. Effects are the addition and subtraction of facts from the system's current model of the world. Examining preconditions and effects leads to another aspect of AI planning systems: the detection and avoidance of conflicts among goals and subgoals. For example, satisfying one goal may produce an effect that prevents the pursuit of an important, related goal or may actually reverse the effects produced by the successful pursuit of earlier goals. Conflicting and cooperative goal interactions are discussed again later in this report.

Similar to PCT, there are different levels at which goals can occur, both higher and lower relative to the goal currently in focus. **Subgoals** support the achievement of that goal and **supergoals** are the goals that motivate it. More generally, subgoals (means) are the ways that supergoals (ends) are achieved.

From these two descriptions, it is clear that there are a number of points on which PCT and AI can agree. They both support the notion of goals, the concept of differences between goal states and current world states, and the need to reduce those differences to bring the (perception of the) world in line with the desired goals. Further, they support the notion of an organised hierarchy of goals. The treatment of preconditions and effects in AI is

P515716.PDF [Page: 23 of 57]

accommodated in PCT through perceptual signals that reflect the nature of the current state of the world and effects that operate on it. An inner PCT loop would allow for rehearsals of reference, perceptual and error signals and is consistent with a reasoning component in AI systems that examines optional subgoal structures that might produce a successful plan. Further elaboration of these inner loops and their relationships within and across levels in PCT would be instructive.

The Notion of Plans

A corresponding and complementary concept to goals in AI is the notion of **plans.** Plans are sets of subgoals (possibly only one) that support the achievement of some higher-level goal. The goal is specified at some level and subgoals are identified that will help achieve that goal. Of course, the goal is a supergoal relative to all the subgoals in the plan but it is referred to simply as a goal with its meaning understood in context.

In PCT, there seems to be some reluctance to embrace the notion of plans either because it seems to imply knowing what may not be knowable (Powers and the Editorial Board of the Control Systems Group, 1993) or that plans somehow reflect brittle, repeated actions that cannot accommodate changes in the real world:

> *"There can be no plan of* ***action*** *precise enough to carry out this process that the driver accomplishes every day. What we really see is not a series of repeated actions that have repeated consequences, but a series of* ***variable*** *actions that have repeated consequences" (Forssell, 1997, p. 9)*

Subgoals seem to be implied, however, in PCT's hierarchical control model. Marken (1993) points to a need for a such a hierarchy based on perceptual dependencies.

> *"The rationale for hierarchical classes of perceptual control is based on the observation that certain types of perception depend on the existence of others. Higher level perceptions depend on (and, thus, are a function of) lower level perceptions." (p. 4)*

Since a reference signal is identified in PCT as synonymous with a goal, it stands to reason that a hierarchy of such reference signals is a hierarchy of goals. Reference signals at lower levels support those higher up and are required to reduce the error signals at higher

levels. Thus, those lower-level signals would seem to constitute ways (plans) for achieving the higher-level signal (goal).

Passing over PCT objections to the use of the word "plan," what seems problematic is how the various reference signals operate in concert with one another to produce intelligent, adaptive behaviour. That is, how is a set of low-level reference signals chosen, from many possible alternate sets, in a way that permits the successful achievement of a higher-level reference signal?

Assuming all of these relationships to be hardwired flies in the face of a concept like adaptation and, for that matter, even claims by PCT theorists themselves. Appealing again to some internal operations (inner loops) could provide a way of linking PCT concepts with adaptation to changing circumstances in one's world.

In many circumstances, goal achievement requires only the application of previously learned behaviour in the presence of familiar circumstances (some world state), monitoring the environment for changes and using other learned methods to adjust behaviour so as to accommodate those changes, as in PCT's familiar example of driving a car to some desired destination.

In contrast, planning an action or activity that is new or that may require substantial variation from past behaviour, demands some kind of (inner loop) rehearsal for the behaviour to be effective. Some form of rehearsal and even practice may be necessary to establish the new behaviour required to achieve the desired goal. Participating low-level behaviours, or at least aspects of those behaviours, may not need to be learned.

To help bridge the gap between PCT and AI approaches on the issue of plans, PCT might be more explicit about the relationships among reference signals in its hierarchically organised systems. Specific examples of adaptive systems created using the PCT approach would be helpful, along with the more general principles and methods used to define relationships within a PCT hierarchy.

The Nature of Hierarchies

A key difficulty in constructing hierarchies is establishing clear criteria for defining their levels. Lower levels (layers) seem more easily identifiable and the criteria for differentiating them easier to specify. Likely, this is due to the fact that "behaviours" at those

levels are far more limited, simpler and easier to identify, detect and measure than those at higher levels. The PCT hierarchy illustrates these points nicely.

Powers (1973; 1990) proposed a hierarchy of eleven levels with the lowest level systems controlling perceptions that represent **intensity** from the environment. A second layer is **sensations**, which includes things like sounds and colours that are functions of intensities at the lowest level. A third level supports the control of **configurations**, which are combinations of sensations and, so it goes, on up the hierarchy.

Descriptions of these low level control systems seem quite reasonable: it seems clear how they support one another and how a system might be constructed based on such a hierarchy. As one ascends the hierarchy, however, things become increasingly tentative and questionable. Of course, this point can be made about many non-PCT hierarchies as well.

At the upper end of Powers' hierarchy are levels called **categories** and **programs**. Categories capture the notion of class membership and **programs** specify if-then contingencies, similar in many ways to computer programs. At these levels, Powers has moved into areas that involve symbolic systems and it is not at all clear why that shift occurs and how criteria used to establish those levels relate to criteria used to differentiate lower levels in his hierarchy. Powers' highest levels of **principles** and **system concepts** refer to very broad notions, namely, generalisable rules and disciplines such as science, mathematics and art.

Two of these higher levels are worth noting, relative to the concepts of goals and plans in AI. The first is a level up from **categories**, called **sequences**. Sequences represent unique orderings of lower order perceptions and provide a first hint at how PCT might be seen to accommodate the notion of plans through logical relations among actions and goals. The next level up from sequences is **programs**, defined in the previous paragraph. The notion of a program captures much of what might be considered a goal and the plan for achieving it, including the specification of contingencies that contribute to the adaptiveness of a plan or to the adoption of a different plan altogether.

Returning to the nature of hierarchies, a little reflection on the levels in Powers' proposed PCT hierarchy will show that it is not at all clear how the higher levels are related in any coherently logical, smoothly transitioning way to the lower levels, nor is it clear whether the criteria used to differentiate any two levels bear any resemblance to those used to distinguish any other pairs. More theoretical work is needed of course, along with implementations and demonstrations of systems that attempt to use the higher levels.

Hierarchies and Plans

To further elaborate the notion of hierarchies and plans, the following example from Hendy, Farrell, and East (2000) is instructive.

> *"Human behaviour is commonly described with action verbs, rather than perceptual states. For instance, going from the first floor of a building to the top floor, one might walk to the elevator, press the door button, enter the elevator, press the numbered button, wait for the doors to close then open, and exit the elevator. This sequence of events might lead the reader to believe that the human is directly controlling their actions, perhaps in response to some stimulus. However, implicit in this elevator scenario is the goal of reaching the top floor, the initial perception of being on the first floor, and a series of actions that move the current perception closer to the goal state."* (p. 7)

Although this example is meant to show that a characterisation using action verbs can be represented as changes in perceptual states, it also provides a convenient way to illustrate how AI might represent those perceptual states in a hierarchy of goals and subgoals (plans). One of many useful rule sets for understanding such examples, developed by Abelson some years ago (see Schank and Abelson, 1977), is formulated as follows:

Actions Cause States;
States Enable Actions

Applying this rule set to the above example, along with the concepts of goal and subgoal (states) and actions (operators), produces the following:

State 1 (Subgoal5):	Being on first floor
Action 1 (Opt1):	Walk to elevator
State 2 (Subgoal4):	Being at elevator
Action 2(Opt2):	Press button for elevator
State 3 (Subgoal3):	Elevator at first floor (and) doors open
Action 3(Opt1):	Walk into (Enter) elevator
State 4 (Subgoal2):	Being in elevator
Action 4(Opt2):	Press button for desired floor
State 5 (Subgoal1):	Elevator at desired floor (and) doors open
Action 5(Opt1):	Walk out of (Exit) elevator
State 6 (**Goal**):	Being on the top floor

If the above states adequately reflect conditions in the real world, then the actions that follow are consistent not only with output functions invoked by the error signals of PCT but also with cognitive theories that require the evaluation of testable conditions to trigger consequent behaviour and even behavioural notions of chained responses to (antecedent controlling) discriminative stimuli.

In AI, the entire sequence can be thought of as a plan for achieving the final goal (State 6: Being on the top floor), but actions at any level in this example can be understood to achieve the goal represented in the following state. Each state is a sub-goal in the service of the final goal (state) and the actions and subgoals together constitute a plan for reaching that final goal, at whatever level that may be defined.

Comparing this to PCT, subgoals are equivalent to controlled real-world variables and thus to perceptual states that occur as consequences of an entity's acting on the world, and from the actions (and properties (states)) of other entities (e.g., the actions and states of the elevator).

As actions achieve each of the subgoals, the final goal (reference signal): being on the top floor is compared against the current world state and an error signal generated which shows that the difference is less than before. That process is repeated as each of the subgoals is achieved until the error signal is effectively zero.

If the plan is a clear one and the currently satisfied condition is a recognisable condition within the plan, the next action to reduce that difference with the final goal should be clear also. If the next "logical" condition is not satisfied, however, replanning may need to take place.

What happens, for example, if the elevator is out of order? Cognitive theories permit replanning by allowing a person to use alternative routes like taking the stairs or asking an attendant for help in getting the elevator working. Conditions in this scenario, which could influence whether and when such replanning takes place, might include a light above the elevator door providing information about where the elevator is and whether it appears to be working.

How might PCT handle the problem of replanning? Will the person determine that the stairs are a best alternative under the circumstances or is he left standing at the elevator repeatedly pushing the button? That is, does the behaviour continue to repeat since the error signal remains the same or does the control system have a way to initiate replanning? An

intriguing question, and one that the reader, in finding an answer, should also uncover other areas of correspondence between AI and PCT.[2]

Hierarchies Revisited

As indicated, there are fuzzy aspects to hierarchies such as deciding the criteria to use to differentiate the different levels, how many there are, and so on. In the example above, the action-state sequence can be thought of as a kind of hierarchy leading to a final goal.

That hierarchy is based primarily on identifying an appropriate sequence of actions and states rather than identifying different levels of abstraction or granularity in those actions and states. In a sense, the whole sequence, as described, can be thought of as existing at one level in an abstraction hierarchy. Within the PCT hierarchy, that likely is the programs level.

To illustrate, contrast the action-sequence hierarchy with one that illustrates any given action in that sequence, e.g., pressing of the elevator button. In order to accomplish that, one has to move the hand, extend a finger and perhaps tilt the body slightly forward as the finger comes into contact with the elevator button. Those acts involve bringing muscles into play, altering muscle cells and, at an even more fundamental level, firing neural impulses. The goal of pushing the button then is achieved by a set of lower-level actions in a different kind of hierarchy than the action-sequence one above, by body movement and position, the flexing of muscles, changes in muscle cells, and activation of neural impulses.

Powers admits that he doesn't know how many levels there are and that he doesn't know what determines the reference signal for the highest level. He points out that there are perhaps thousands or even more control systems, e.g., one or more for each of the 800 muscles in the body. He also recognises that some of those systems may operate independently and simultaneously, or even dependently.

Much of the understanding of hierarchical representations for the design and implementation of adaptive systems is yet to come. Other forms of representation are also possible and should be considered for what they have to say to hierarchical systems such as those in the PCT and AI fields.

---

[2] Interestingly, reports on some of Wolfgang Koehler's early experiments in Gestalt Psychology describe how some animals, like chickens, will try to reach food they can see through a barrier by repeatedly crashing into the barrier. They are unable to figure out how to go around the barrier to reach the food, as higher-order animals can do.

Shared Concepts for Intelligent Adaptive Interfaces

Only a few of the concepts from PCT and AI that continue to be important in the analysis, design and implementation of intelligent adaptive interfaces have been explored here. This discussion is not meant to be a thorough review and comparison of the two areas, only a primer on some of the relations among their concepts.

Many other aspects of PCT and AI could have been discussed such as how each might handle conflict and cooperation, within and among sets of control systems (people and machines), or how they might account for the modelling of one set of control structures by another in the service of understanding what goals (reference signals) are appropriate in interpersonal situations (see for example Edwards and Sinclair, 2000).

Classification of goal relationships from AI might motivate an exploration of how PCT can accommodate these to build adaptive control systems. The following is a modification from Wilensky (1983), and shows one way to classify goal relationships:

	Negative Interactions	**Positive Interactions**
Internal	Conflict: Mutually opposing goals held by a single entity (person or machine)	Overlap: Goals achieved more easily together than apart.
External	Competition: Mutually opposing goals held by different entities (people or machines)	Concord: Mutually beneficial goal possessed by several entities.

Table 1. Goal Interactions

Competition, cooperation and coordination of these different relationships among goals, within and across entities, needs to be accommodated within PCT and it will be instructive to explore just how PCT deals with each.

Finally, comments and questions raised here are not meant to argue in favour of either PCT or AI in building intelligent adaptive systems, but rather to encourage the elaboration of ideas within both so that correspondences and differences can be understood more clearly. Achieving that goal will provide real opportunities for mutually beneficial discussion and effective refinement and extension of adaptive system building.

Generalising Plan Recognition Principles to Other Projects at DCIEM

Projects at DCIEM that stand to benefit from the intelligence testbed work being conducted with the *LOCATE* Workspace Layout Tool include the two related projects on the Integrated Performance Modelling Environment (IPME) and Safework®.

Since *LOCATE* is seen as one tool in a suite of tools, the near-term target applications are the others that make up that suite: Safework®, which is a computer-based three-dimensional man model, and IPME, a mission, function and task analysis tool used to model human behaviour and predict performance.

Integrated Performance Modelling Environment (IPME)

This is a tool for conducting front-end human engineering and human performance modelling for validation and verification analysis.

Early versions were developed with funding from the Department of Defence in both the United Kingdom and Canada, the latter through DCIEM. It is now commercially available from Micro Analysis and Design in the US. Current work aims to integrate features of a closely related application called, SOLE (System Operator Loading Evaluation) into the IPME product.

As summarised in Greenley (1999), SOLE IPME is used:

> *"to conduct mission/function/ task analysis, to model human behaviour, and to predict human performance under future operational conditions.*
>
> ....
>
> *It focuses on the simulation of human operators in their operational environments. IPME allows the user to construct component models that are tied together in a simulation environment. [It] has five component models, a measurement suite that can be used for blocked design of experiments to evaluate human performance or the effects of system changes on human performance, and a number of operator workload measurement methods." (p. 25)*

P515716.PDF [Page: 31 of 57]

More details concerning IPME may be found on the web at:

http://199.170.148.19/toolsite/Tools/Shrtdesc/sindipme.htm

Safework®

This is a commercially available, computer-based three-dimensional man-model based on anthropometric data from US and Canadian surveys. It was developed for DCIEM by Genicom Consultants of Montreal.

Safework® provides an accurate model of humans that takes into account gender, ethnic origin and an additional 104 anthropometric variables. Application domains include "interior vehicle design, workspace design, product design and prototyping, process simulation and ergonomic analysis." (Greenley, 1999, p. 29)

The long-term vision is the combination of *LOCATE*, IPME and Safework® into a tightly integrated suite of mutually complementary tools. One of the goals of that integration is to provide intelligent, adaptive aiding for its users. A common interface is a basic requirement for all of those tools as well as a consistent system that will support users in learning and using the entire suite.

Whereas the other two tools provide some standard help support for their users, they do not contain any support for intelligent aiding. Adding intelligent aiding capabilities to *LOCATE* was a first step in the direction of providing adaptive help for the broader suite of software tools. Transforming that work into a testbed for the study of intelligent adaptive aiding was the second step.

As the work on the testbed continues to mature, efforts will be directed at interfacing the emerging intelligent aiding capabilities with the IPME and Safework® tools. A first step in that direction for AIM has been the familiarisation with the intent and functionality of those tools.

Incorporating Planning Into a Standalone Pluggable Intelligent Aiding Module

This is an ambitious task and can be accomplished only in a staged manner. Key requirements for a pluggable intelligent aiding system are emerging from the work on *LOCATE* but many requirements for a successful system may rely on the software applications in which they are to be embedded.

The first task, and one that has been accomplished, is to isolate the code in *LOCATE* that handles the tracking of user interface actions, the matching of those actions to a graph of possible goals and plans and the display of those results in various model windows. In addition, the inferences from those results and the consequent intelligent, adaptive aiding provided to the user are also part of that isolated code.

The next step is to generalise those procedures so that they can be applied to other applications. Some of that work has begun and was described in earlier sections. It includes the construction of a hierarchy of actions, goals and plans for matching to a user's interface actions, and the recent alternative approach of goal and plan recipes, which provide components that may be more easily adapted to other application domains.

The work on generalisation serves two purposes: developing procedures which are portable to other domains and providing a basis for comparison of plan recognition and plan generation techniques so that issues like elegance, performance and portability can be addressed directly for particular applications. This is a key advantage to the testbed work that is currently being conducted with *LOCATE*.

To make the process of generalising procedures and porting intelligent aiding techniques to other applications, work is envisioned on a **interface analyser**. The purpose of the analyser will be to characterise the interface elements and program functionality of target applications for the purpose of facilitating the match between the application and the procedures of intelligent aiding developed within the testbed.

The development of the analyser should occur in three stages. The first stage involves developing an understanding about how the software might address questions that need to be asked about target applications, including 1) its overall purpose; 2) general features of the software that support that purpose; 3) the relationship between interface

actions and the goals and subgoals users are trying to achieve when applying the software to problems in the application domain; 4) how those actions, goals and plans can be organised into a coherent hierarchy of relationships; 5) the scope of interface support for those actions; and, 6) the help requirements of application users.

This understanding will be developed in concert with application developers and expert users. At this first stage of development, the analyser will be a knowledge acquisition tool and much of the data will be obtained by humans working with other humans guided by the queries from the analyser software.

The second stage in building an interface analyser will be to automate part of the process of acquiring that information. It is clear that the requirements are such that the analyser would have to be an expert in the problem domain to acquire all the data necessary to customise intelligent aiding for a target application. As this is impractical, the focus will be on what information applications need to "publish" about their functionality and interface support, for the analyser to accomplish its task. Further, requirements should be identified for how the analyser is to use that information in customising a set of intelligent aiding techniques to the target application.

The third stage of development will be an extension and refinement of the analyser as experience is gained with its use. That process will force issues concerning the generalisability of adaptive techniques as well as information requirements for what applications need to publish about themselves to support such generalisation. The latter could be useful to a wide variety of systems that might be embeded within a target application. It could also serve a variety of communication goals among heterogeneous software that need to" talk" to each other.

Demonstrations of an interface analyser tool will be important both for its development and for its generalisability to other applications. An obvious first step will be to create a version of *LOCATE* that has been divorced from its intelligent aiding capabilities but that will provide information necessary for the analyser to re-fit the intelligent help system to the basic vesion of the software. That work will involve deciding what information *LOCATE* needs to publish to allow the analyser to perform such a re-fit and incorporating procedures by which *LOCATE* can provide that information.

Working with *LOCATE* will be an important step to generalising the requirements and procedures to other applications. As the work on *LOCATE* proceeds, the information requirements will be extended to the other two tools in the suite of tools of which *LOCATE* is

a part, i. e., Safework® and IPME. Eventually, information about the goals and purposes of the three applications, along with information about the interface support for those purposes, will be made available to the analyser.

Demonstrations could then be conducted showing how, given published information by each of the three applications as to their purposes and interface support, the interface analyser is able to customise intelligent aiding for each.

Although full automation may not be possible for some time and may depend on how useful such a process is perceived by the software development community, the work should provide important insights into the requirements for automating the process of adapting intelligent aiding to many software applications. This should be true both for the information needed from applications and the procedures required for using that information to adapt and incorporate intelligent support.

P515716.PDF [Page: 35 of 57]

References

Bass, E.J., Ernst-Fortin, S.T., and Small, R.J. (1997). Knowledge base development tool requirements for an intelligent monitoring aid. *Proceedings of the Tenth Annual Florida Artificial Research Symposium (FLAIRS-97)*, Daytona Beach, FL, May 12-14, pp. 412-416.

Broadbent, G. (1988). *Design in architecture.* London: David Fulton Publishers.

Edwards, J. L. & Mason, J. A. (1988). Toward intelligent dialogue with ISIS. *International Journal of Man-Machine Studies, 28*, 309-342.

Edwards, J. L., & Hendy, K. (1992). Development and validation of user models in an air traffic control simulation. Paper presented at the Second International Workshop on User Modelling. International Conference and Research Center for Computer Science (IBFI), Dagstuhl Castle, Germany, August 10-13, 1992.

Edwards, J. L., & Hendy, K. (1999). A Testbed for Intelligent Aiding in Adaptive Interfaces. Paper presented at the 1999 AAAI Spring Symposium on Adaptive User Interfaces, Stanford University, March 20-22.

Edwards, J. L., & Sinclair, D. (2000). Designing intelligence: A case of explicit models and layered protocols. In M. M. Taylor, F. Néel, and D. G. Bouwhuis (Eds.), *The structure of multimodal dialogue II.* Philadelphia, PA: John Benjamins.

Edwards, J. L., Scott, G.L, & Hendy, K. (in preparation). The LOCATE Workspace Layout Tool as a Basis for an Intelligence Testbed.

Forssell , Dag (1997, update) PCT introduction and resource guide. CSG group materials online. http://www.ed.uiuc.edu/csg/

Greenley, M. P. (1999). The "Way Ahead" for Human Factors Engineering Tools. A report prepared for the Canadian Department of National Defence, PWGSC Contract No. W7711-8-7464 by Greenley & Associates Incorporated, Waterloo, ON, Canada

Hendy, K. C. (1984). 'Locate': A program for computer-aided workspace layout. Master's Thesis, Department of Electrical Engineering, Monash University, Clayton, Victoria, Australia.

Hendy, K. C. (1989). A Model for Human-Machine-Human Interaction in Workspace Layout Problems. *Human Factors, 31(5),* 593-610.

Polson, M.C., & Richardson, J.J. (Eds.) (1988). *Foundations of intelligent tutoring systems.* Hillsdale, N.J.: Lawrence Erlbaum.

Marken, R. S. (1993). The hierarchical behaviour of perception. Closed Loop, *Journal of Living Control Systems, 3(4).*

P515716.PDF [Page: 36 of 57]

Newell, A., & Simon, H. (1963). GPS: A program that simulates human thought. In E. A. Feigenbaum and J. A. Feldman (Eds.), *Computers and thought.* New York.: McGraw-Hill.

Newell, A., & Simon, H. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice-Hall.

Powers, W. T. (1973) *Behaviour: The control of perception.* New York: Aldine De Gruyter.

Powers, W. T. (1990) A hierarchy of control. In R. J. Robertson and W. T. Powers (Eds.), *Introduction to modern psychology: The control- theory view.* Gravel Switch, KY.: The Control Systems Group Inc..

Powers, W.T. (1999). A brief introduction to perceptual control theory. Control Systems Group: Bill T. Powers Home Page: http://www.frontier.net/~powers_w/whatpct.html.

Powers, W. T., & the Editorial Board of the Control Systems Group (1993). The dispute over control theory. Paper presented at CSG meeting, Durango, CO, July-August.

Richardson, R. J., & Powers, W. T. (Eds.) (1990) *Introduction to modern psychology: The control-theory view.* Gravel Switch, KY: The Control Systems Group.

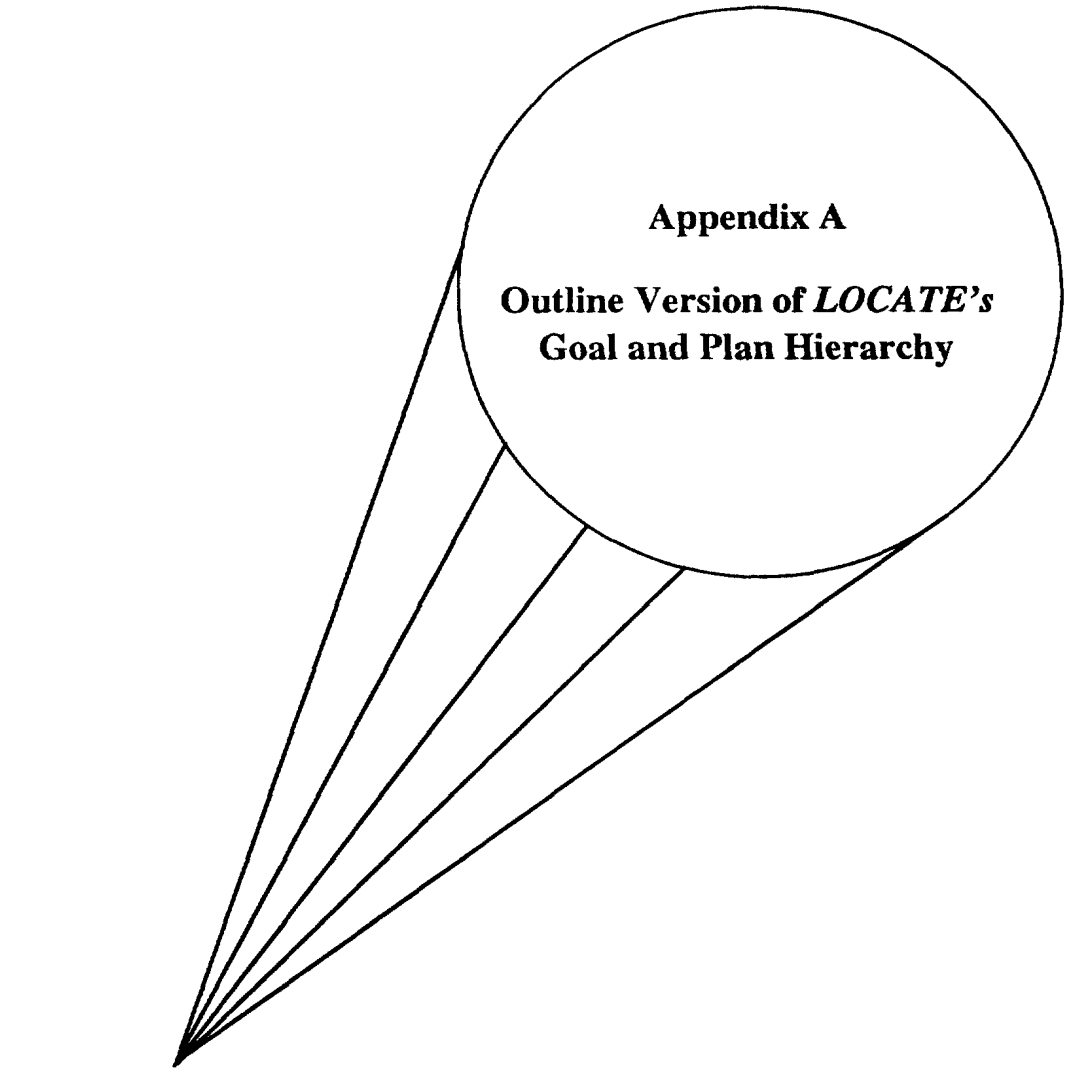
Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals and understanding.* Hillsdale, N. J.: Lawreence Erlbaum Associates.

Simon, H. (1981). *The sciences of the artificial.* 2nd edition. Cambridge, MA: MIT Press.

P515716.PDF [Page: 37 of 57]

Appendix A

Outline Version of *LOCATE's* Goal and Plan Hierarchy



Artificial Intelligence
Management and Development Corporation

P515716.PDF [Page: 38 of 57]

The following is the outline form of *LOCATE's* plan hierarchy.

To work with a design

To produce an optimised design

To create a design

To examine the attributes of the Workspace X (x1, x2,...,xn)

To open the "Dimensions" portion of the WS attributes window

To select "Workspace: from the View Menu

To open the LOCATE application

To double-click in an empty portion of the WS.

To open the LOCATE application

To open the "Domain Weights" portion of the WS attributes window

To select Domain Weights from the View Menu

To open the LOCATE application

To click the "Domain Weights" tab

To open the "Dimensions" portion of the WS attributes window

To open the "Object Overlap" portion of the WS attributes window

To click the "Object Overlap" tab

To open the "Dimensions" portion of the WS attributes window

To open the "Domain Weights" portion of the WS attributes window

To open the "Object Count" portion of the WS attributes window

To click the "Object Count" tab

To open the "Domain Weights" portion of the WS attributes window

To open the "Dimensions" portion of the WS attributes window

P515716.PDF [Page: 39 of 57]

To change the attributes of the Workspace X (x1, x2,...,xn)
To change the x-dimension of the workspace
To click OK or Apply in the WS attributes window
To change the value in the x-dimension text box
To open the Dimensions portion of the WS attributes window
To change the y-dimension of the workspace
To click OK or Apply in the WS attributes window
To change the value in the y-dimension text box
To open the Dimensions portion of the WS attributes window
To change the x-component of the zero point of the workspace
To click OK or Apply in the WS attributes window
To change the value in the zero point x-component text box
To open the Dimensions portion of the WS attributes window
To click on a zero point radio button
To open the Dimensions portion of the WS attributes window
To change the y-component of the zero point of the workspace
To click OK or Apply in the WS attributes window
To change the value in the zero point y-component text box
To open the Dimensions portion of the WS attributes window
To click on a zero point radio button
To open the Dimensions portion of the WS attributes window
To change the maximum workspace dimension of the workspace
To click OK or Apply in the WS attributes window
To change the value in the max. WS dimension text box
To open the Dimensions portion of the WS

attributes window

To change the circular radius of the workspace

To click OK or Apply in the WS attributes window

To change the value in the circular radius text box

To open the Dimensions portion of the WS attributes window

To change the ruler scale of the workspace

To click OK or Apply in the WS attributes window

To change the value in the ruler scale text box

To open the Dimensions portion of the WS attributes window

To change the number of decimals of the workspace

To click OK or Apply in the WS attributes window

To change the value in the number of decimals text box

To open the Dimensions portion of the WS attributes window

To change the zoom factor of the workspace

To click OK or Apply in the WS attributes window

To change the value in the zoom factor text box

To open the Dimensions portion of the WS attributes window

To change the length to breadth ratio of the workspace

To click OK or Apply in the WS attributes window

To change the value in the length to breadth ratio text box

To open the Dimensions portion of the WS attributes window

To change the first order link weight of the workspace

To click OK or Apply in the WS attributes window

To change the value in the first order link weight text box

To open the Domain Weights portion of the WS attributes window

To click the first order link weight text box

To open the Domain Weights portion of the WS attributes window

P515716.PDF [Page: 41 of 57]

To change the auditory link weight of the workspace
To click OK or Apply in the WS attributes window
To change the value in the auditory link weight text box
To open the Domain Weights portion of the WS attributes window
To click the auditory link weight checkbox
To open the Domain Weights portion of the WS attributes window
To change the distance link weight of the workspace
To click OK or Apply in the WS attributes window
To change the value in the distance link weight text box
To open the Domain Weights portion of the WS attributes window
To click the distance link weight checkbox
To open the Domain Weights portion of the WS attributes window
To change the tactile link weight of the workspace
To click OK or Apply in the WS attributes window
To change the value in the tactile link weight text box
To open the Domain Weights portion of the WS attributes window
To click the tactile link weight checkbox
To open the Domain Weights portion of the WS attributes window
To change the visual link weight of the workspace
To click OK or Apply in the WS attributes window
To change the value in the visual link weight text box
To open the Domain Weights portion of the WS attributes window
To click the visual link weight checkbox
To open the Domain Weights portion of the WS attributes window
To create an object X
To click in the WS
To select tool T in tool palette

P515716.PDF [Page: 42 of 57]

To open the LOCATE application
To click-hold-and drag in the WS
To select tool T in tool palette
To create N instances of object X (N > 1)
To click N times in the WS
To double-click on tool T in palette
To click OK in the Multi-Object Creation window
To enter the number of instances N
To shift-click on tool T in palette
To change the attributes of object X
To open the attributes window of object X
To create object X
To select object X
To create object X
To select object X in name pop-up menu in object Y's attributes window
To open the attributes window of object Y
[Optional] To select the attributes tabbed section in an object's window
To open an object's Link Functions Window
To create object X
To select object X
To open an object's Priority Weights Window
To create object X
To select object X
To change attributes with mouse by click and drag (move; resize; rotate)
To move object X
To select object X and drag
To resize object X
To click on a handle of object X and drag
To select object x
To rotate object X
To click and drag in a circular fashion
To select "Rotate" from the Arrange Menu
To select object x

P515716.PDF [Page: 43 of 57]

To configure an object Y (where, Y=EW; EOb; FOb)
    To create an object X inside Y (where, X≠Y)
        To create an object Y (where, Y=EW; EOb; FOb)
    To move an existing object X (where, X≠Y) inside Y.
        To click and drag object X over the top of object Y
            To select object X (where, X≠Y)
                To create an object Y (where, Y=EW; EOb; FOb)
To delete an object X
    To press the delete key
        To select an object X
    To select "Cut" in the Edit menu
        To select an object X
    To select "Clear" in the Edit menu
        To select an object X
To delete the contents of object X (where, X=EW; EOb; FOb)
    To delete an object Y
        To select an object Y inside the object X
To group objects (x1, x2,...,xn)
    To select "Group" from the Arrange menu
        To select objects (x1, x2,...,xn)
To Ungroup objects (x1, x2,...,xn)
    To select "Ungroup" from the Arrange menu
        To select grouped object X
To duplicate an object X
    To select "Duplicate" from the Edit menu
        To select an object X
    To select "Paste" from the Edit menu
        To select "Copy" from the Edit menu
            To select an object X
To select all objects
    To select "Select All" from the Edit Menu
To send an object X to the back
    To select "Send to Back" from the Arrange menu
        To select an object X

P515716.PDF [Page: 44 of 57]

To send an object X backward
To select "Send Backward" from the Arrange menu
To select an object X
To bring an object X forward
To select "Bring Forward" from the Arrange menu
To select an object X
To bring an object X to the front
To select "Bring to Front" from the Arrange menu
To select an object X

To modify an existing design

To open a LOCATE design
To select "Open" in the standard OPEN dialox
To locate and select the file to be opened.
To select "Open" from the File Menu
To import a DXF File (design)
To specify "Import Options"
To select "Open" in the standard OPEN dialox
To locate and select the file to be imported.
To select "Import --> DXF" from the File Menu.

To analyse a design.

To run a cost function
To (manually) run a Cost Function
To click on the cost function value button
To select "Cost Function" from the Execute menu
To instruct LOCATE to run (automatically) a cost function
To select the "automatic" radio button
To select "Cost Function" from the Execute menu
To run the LOCATE optimiser
To press the "Optimise" button
[Optional] To enter a distance step size
[Optional] To enter an angular step size
To select "Optimise" from the Execute menu

P515716.PDF [Page: 45 of 57]

To preserve a design

To save a design

To save a design as a LOCATE design

To select "Save" from the File Menu

To Click "OK" in the Standard SAVE dialox.

To specify the [accept default] name of file to be saved.

To select "Save As..." from the File Menu

To export a design as a DXF file (design)

To Click "OK" in the Export dialox.

To specify the [accept default] name of file to be exported.

To select "Export..." from the File Menu

To print

To modify Page Setup attributes

To click "OK" in the Page Setup dialox

To specify Page Setup attributes

To select "Page Setup" from the File Menu.

To print

To print a design

To click "OK" ["Print"] in the Print dialox

To specify print options

To select "Print" from the File Menu.

To print a colour cost display

To click "OK" ["Print"] in the Print dialox

To specify print options

To click the "Print" button on a cost display window.

To select "Cost Display" from the View Menu

To print cost function values

To click "OK" ["Print"] in the Print dialox

To specify print options

To click the "Print" button on the cost function window.

To click the "Run Again" button on the cost

P515716.PDF [Page: 46 of 57]

function window
[Optional] To check "Display Details"
[Optional] To check "Add Details to Output File
To select "Cost Function..." from the Execute Menu

To suspend work on a design

To recall another design from the Cost Function History
To open another design
To make another application active.
To close the LOCATE application
To select "Quit" from the File Menu [Alternates]

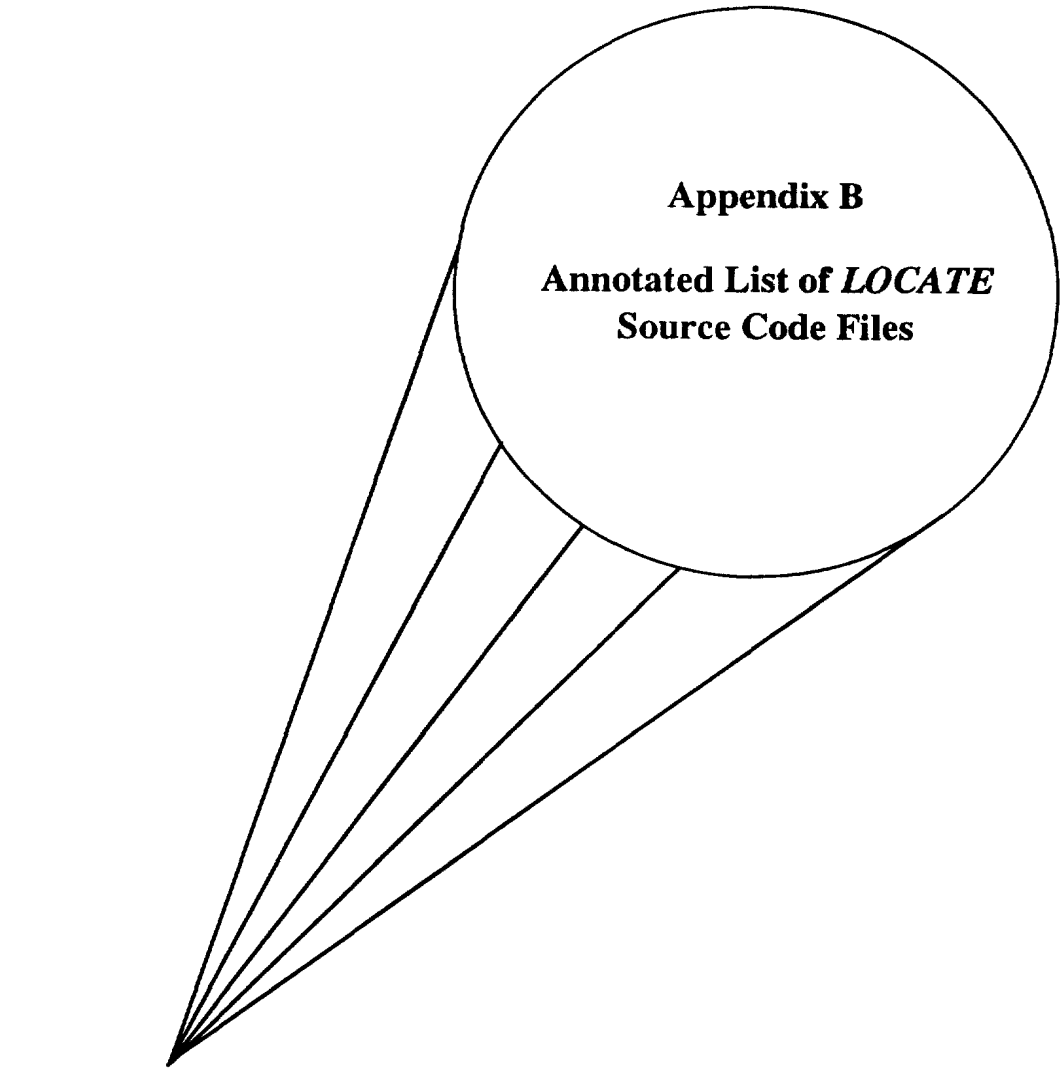
To abandon a design

To delete a design
To delete all objects in design X
To select "Clear" from the File Menu
To select all objects in design X
To select "Cut" from the File Menu
To select all objects in design X
To type the Backspace or Delete key
To select all objects in design X
To delete the design file from the drive on which it is stored.

P515716.PDF [Page: 47 of 57]

Appendix B

Annotated List of *LOCATE* Source Code Files



Artificial Intelligence
Management and Development Corporation

P515716.PDF [Page: 48 of 57]

Notes:

i) .c files are C source code
.h files are C header files
.rc files are Open Interface resource description files

ii) Some extraneous files in the *LOCATE* folders are outdated and will be erased in future.

AEVENT.C

- This file contains the code to handle Apple Events for opening documents by double-clicking in the Finder.

ALLOBJINFO.C, ALLOBJINFO.RC

- The "AllObjInfo" module contains the code and Open Interface resources necessary for the "All Objects Info" window.

AOBS.C, AOBS.RC

- The "AObs" module contains the code and Open Interface resources necessary for the "Fixed Obstruction" window.

ASSIGN.C

- Original *LOCATE* C file

CFALERT.C, CFALERT.RC

- The "CFAlert" module contains the code and Open Interface resources necessary for the alert box that appears when there have been changes to the design but no cost function has been run since those changes have been made. The alert box appears before displaying the Cost Function History window .

CFBROWSE.C, CFBROWSE.RC

- The "CFBrowse" module contains the code and Open Interface resources necessary for the "Cost Function History" window.

CFCHECK.C, CFCHECK.RC

- The "CFCheck" module contains the code and Open Interface resources necessary for the "Cost Function Checks" window.

COSTCOLR.C, COSTCOLR.H, COSTCOLR.RC

- The "CostColr" module contains the code and Open Interface resources necessary for the "Cost Display Editor" window.

COSTDISP.C, COSTDISP.H, COSTDISP.RC

- The "CostDisp" module contains the code and Open Interface resources necessary for the "Cost Display" window.

P515716.PDF [Page: 49 of 57]

COSTFN.C, COSTFN.RC
- The "CostFn" module contains the code and Open Interface resources necessary for the "Cost Function" window.

DRAWROTD.C, DRAWROTD.H
- C code for handling the drawing of rotated objects

DXF.H
- Header file with DXF format constants

DXFOPT.C, DXFOPT.H
- The "DXFOpt" module contains the code and Open Interface resources necessary for the "DXF Import Options" window.

EDITOR.C, EDITOR2.C, EDITOR.RC
- The "Editor" module contains the code and Open Interface resources necessary for the main *LOCATE* window (includes code for Diagrammer, palette, rulers).

EVAL1.C
- Original *LOCATE* C file

EWATTR.C, EWATTR.H, EWATTR.RC
- The "EWAttr" module contains the code and Open Interface resources necessary for the "Workstation" window, which now includes separate tabbed sections for Attributes, Link Functions and Priority Weights.

EXTERN.H
- Original *LOCATE* header file

FORMAT.H
- Original *LOCATE* header file

FUNCT1.C
- Original *LOCATE* C file

GOALOBJ.CPP
- Contains code for defining and handling the C++ goal object

HEADER.DXF
- Contains information that gets added to all exported DXF files

HELPALRT.C, HELPALRT.RC
- The "HelpAlrt" module contains the code and Open Interface resources necessary for displaying the "Help Message" windows.

P515716.PDF [Page: 50 of 57]

HELPMORE.C, HELPMORE.RC
- The "HelpMore" module contains the code and Open Interface resources necessary for displaying the "Help Message" windows containing the "More" button.

IMPRTDXF.C
- C code for handling the importing of a workspace from DXF format

INFOUPD.C, INFOUPD.H, INFOUPD.RC
- The "InfoUpd" module contains the code and Open Interface resources necessary for the "Information Update" window.

INFOWIN.C
- C code for handling the "Object Info" window

LFSUMM.C, LFSUMM.RC
- The "LFSumm" module contains the code and Open Interface resources necessary for the "Link Function Summary" window.

LINKDISP.C, LINKDISP.RC
- The "LinkDisp" module contains the code and Open Interface resources necessary for the "Link Display" window.

LOCATE.C
- Based on the original *LOCATE*.C file, this contains the code necessary for loading in a workspace and for computing the cost function.

LOCATE.DAT
- Open Interface compiled resources that are used by *LOCATE* at run-time.

LOCATE
- The *LOCATE* application

LOCATE.H
- Original *LOCATE* header file

LOCATE.μ
- *LOCATE* project for CodeWarrior 11

LOCATE.RC
- Open Interface resources in text format

LOCNEW.C
- C code for handling the creation of a new workspace

LOCNEWEW.C
- C code for handling the creation and deletion of workstations and obstructions

P515716.PDF [Page: 51 of 57]

LOCSAVE.C
- C code for handling the saving of a workspace

LOCSAVEDXF.C
- C code for handling the saving of a workspace in DXF format

MAIN.C, MAIN.RC
- The "Main" module contains the "main" function which starts up the application.

MISC.C
- Original *LOCATE* C file

MISCRSRC.RC
- The "MiscRsrc" module contains Open Interface resources needed by the application (primarily menu and icon resources).

MULTIOBJ.C, MULTIOBJ.RC
- The "MultiObj" module contains the code and Open Interface resources necessary for the "Multiple Object Creation" window.

NEWUSER.C, NEWUSER.H, NEWUSER.RC
- The "NewUser" module contains the code and Open Interface resources necessary for the "About You" window.

OPT.C
- Optimizer code for changing positions and angles

OPTIM.C
- Original *LOCATE* C file

OPTOPT.C, OPTOPT.RC
- The "OptOpt" module contains the code and Open Interface resources necessary for the "Optimizer Options" window.

OPTSET.C, OPTSET.RC
- The "OptSet" module contains the code and Open Interface resources necessary for the "Optimizer Settings" window.

OPTSTAT.C, OPTSTAT.RC
- The "OptStat" module contains the code and Open Interface resources necessary for the "Optimizer Status" window.

OPTSWAP.C
- Optimizer code for swapping workstations

P515716.PDF [Page: 52 of 57]

ORIGIN.C
- Original *LOCATE* C file

OTHEROBJ.C, OTHEROBJ.RC
- The "OtherObj" module contains the code and Open Interface resources necessary for the "Other Object" window.

OUTPUT.C
- Original *LOCATE* C file

PALEDIT.C, PALEDIT.RC
- The "PalEdit" module contains the code and Open Interface resources necessary for the "Palette Editor" window.

PLANS.C, PLANS.RC
- The "Plans" module contains the code and Open Interface resources necessary for the "Plans" window.

PLANREC.C
- The "PlanRec" module contains the code that creates the plan fragments and performs plan recognition.

PRINTPREV.C, PRINTPREV.RC
- The "PrintPrev" module contains the code and Open Interface resources necessary for the "Print Preview" window.

RULEEW.C, RULEEW.H, RULEEW.RC
- The "RuleEW" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Workstation attributes.

RULEGEN.C, RULEGEN.H, RULEGEN.RC
- The "RuleGen" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up object attributes.

RULEOB.C, RULEOB.H, RULEOB.RC
- The "RuleOB" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Obstruction attributes.

RULEOO.C, RULEOO.H, RULEOO.RC
- The "RuleOO" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Other Object attributes.

P515716.PDF [Page: 53 of 57]

RULER.C, RULER.RC

- The "Ruler" module contains the code and Open Interface resources necessary for the "Ruler" window.

SMRTHELP.C, SMRTHELP.RC

- The "SmrtHelp" module contains the code and Open Interface resources necessary for the "Smart Help" window.

SPLASH.C, SPLASH.RC

- The "Splash" module contains the code and Open Interface resources necessary for the startup screen.

SPLASH2.C, SPLASH2.H, SPLASH2.RC

- The "Splash2" module contains the code and Open Interface resources necessary for the "More on *LOCATE*" window.

START.C, START.RC

- The "Start" module contains the code and Open Interface resources necessary for the usability "Start" window.

STARTUP.C, STARTUP.RC

- The "Startup" module contains the code and Open Interface resources necessary for the help reminder at startup.

STARTLOG.C, STARTLOG.RC

- The "StartLog" module contains the code and Open Interface resources necessary for the startup window that allows the user to enter a user name for the help system.

SYSMODL.C, SYSMODL.RC

- The "SysModl" module contains the code and Open Interface resources necessary for the "System Model" window.

TASKMODL.C, TASKMODL.RC

- The "TaskModl" module contains the code and Open Interface resources necessary for the "Task Model" window.

TREE.C

- The "Tree" module contains the code that constructs and searches the plan hierarchy.

USERMODL.C, USERMODL.RC

- The "UserModl" module contains the code and Open Interface resources necessary for the "User Model" window.

WEBBROWS.C, WEBBROWS.H, WEBBROWS.RC

- The "WebBrows" module contains the code and Open Interface resources necessary for the "Web Browser" window.

P515716.PDF [Page: 54 of 57]

WOBS.C, WOBS.RC

- The "WObs" module contains the code and Open Interface resources necessary for the "Elemental Obstruction" window.

WSATTR.C, WSATTR.RC

- The "WSAttr" module contains the code and Open Interface resources necessary for the "Workspace Attributes" window.

DOCUMENT CONTROL DATA SHEET

1a PERFORMING AGENCY	2. SECURITY CLASSIFICATION
Artificial Intelligence Management and Development Corporation, 206 Keewatin Avenue, Toronto, ON M4P 1Z8 CANADA	UNCLASSIFIED -
1b PUBLISHING AGENCY DCIEM	

3. TITLE

(U) Plans and Goals in a Testbed for Intelligent Aiding - Final Report

4. AUTHORS

Edwards, Jack L.

5 DATE OF PUBLICATION	6 NO. OF PAGES
July 1 , 2000	47

7. DESCRIPTIVE NOTES

8. SPONSORING/MONITORING/CONTRACTING/TASKING AGENCY
Sponsoring Agency:
Monitoring Agency:
Contracting Agency : DCIEM
Tasking Agency:

9. ORIGINATORS DOCUMENT NUMBER	10 CONTRACT GRANT AND/OR PROJECT NO.	11. OTHER DOCUMENT NOS
Contract Report CR 2001-011	PW&GS W7711-8-7564/001/TOR; 3ad13	AC203

12. DOCUMENT RELEASABILITY

Unlimited distribution

13. DOCUMENT ANNOUNCEMENT

Unlimited

14 ABSTRACT

(U) The work of this contract extends efforts on an intelligence testbed using the LOCATE Workspace Layout Tool Work on plan recognition aspects of the testbed was extended in two ways first, by creating an action, goal and plan hierarchy that permits LOCATE to match user actions to possible hierarchies, and second, by developing a generalisation of the goal and plan hierarchy through a decomposition into two-level, hierarchical recipes, which can be mixed and matched from various actions at the interface

Other work included comparing concepts used in the AI planning work in LOCATE to similar concepts in an Information Processing/Perceptual Control Theory (IP/PCT) model being developed at DCIEM, similarities and differences of the two approaches were identified and discussed. Ideas for generalising LOCATE's planning principles to development projects such as IPME® and Safework®, projects supported by DCIEM, were explored Finally, the concept of building an interface analyser, to help automate customisation of intelligent aiding for other applications, was described and discussed.

Le travail régi par le présent contrat continue les efforts fournis pour un banc d'essai d'intelligence utilisant l'outil d'aménagement de l'espace de travail, LOCATE Les travaux couvrant les aspects de reconnaissance de plan touchant le banc d'essai ont été accrus de deux manières : premièrement, en établissant une hiérarchie d'action, de but et de plan, qui permet à LOCATE de jumeler les actions des utilisateurs à des hiérarchies possibles et, deuxièmement, en procédant à une généralisation de la hiérarchie de plan et de but par une division hiérarchique à deux niveaux des recettes, qui permettent un choix de combinaisons générées par différentes actions à l'interface.

Notre travail consistait entre autres à comparer les concepts utilisés dans le travail de planification IA de LOCATE avec des concepts similaires dans un modèle de traitement de l'information/théorie perceptuelle de commande (TI/TPC) qui est en train d'être mis au point à l'IMED Les similarités et les différences des deux approches ont été déterminées et ont fait l'objet de discussion. On a étudié à fond les idées qui visent à appliquer les principes de planification du LOCATE à l'ensemble de projets tels que l'IPME® et le Safework®, projets qui sont appuyés par l'IMED. Enfin, l'idée de construire un analyseur servant à personnaliser l'aide intelligente pour les autres applications a été décrite et a fait l'objet de discussion.

15. KEYWORDS, DESCRIPTORS or IDENTIFIERS

(U) HUMAN ENGINEERING TOOLS; HUMAN MODELLING; WORKSPACE LAYOUT; WORKSPACE DESIGN; FACILITY LAYOUT; COMPUTER-AIDED DESIGN; INTELLIGENT HELP; ADAPTIVE INTERFACES; LOCATE; EXPLICIT USER MODELS

515716

CA011071